第1章 ようこそFPGA/ASIC設計の世界へ!

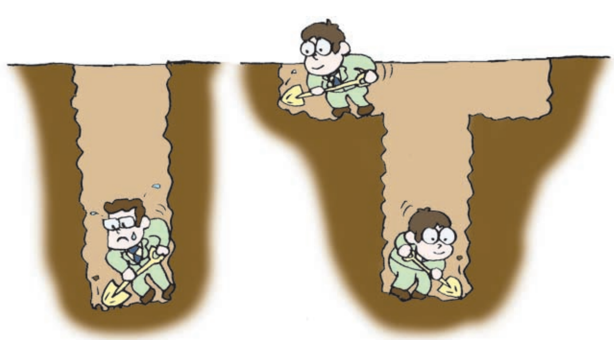
――エンジニアとしての心構えと身に付けるべき技術

浅井 剛

本稿では，今年新たにLSI設計の世界へ入ろうとしているフレッシャーズのみなさんを対象に，みなさんを迎える側にいる技術者からのメッセージをお届けします．FPGA/ASIC設計を行うにあたっての心構えを，筆者の経験をもとにまとめました．LSI設計の世界で，みなさんがより大きくはばたくことを期待しています．（筆者）

「最近の若者は…」とか「今年の新入社員は○○タイプ」と，世代ごとひとまとめにした言いかたがあります．筆者も入社時代には言われ，今は言う側になりました．歳を重ねるにつれ，このことばは単純な嘆きではないと考えるようになりました．若者の成長に期待する一方で，決して追い越されまいと自分自身を叱咤激励するために使っているのだと自覚しています．賛否はあると思いますが，いろいろな意味でこのことばはずっと使われ続けるでしょう．

毎年，希望に目を輝かせた新入社員を見るこの季節，早くしごとに慣れて，一人前の技術者になってほしいと願っています．



（a）I型技術者　（b）T型技術者

図1　I型技術者とT型技術者

（a）のI型技術者は，自分が担当している分野ではほかを寄せ付けないほど非常に深い知識を持っている技術者のことである．（b）のT型技術者は，自分が担当している分野だけでなく，その周辺にある技術に対しても幅広い知識を備えた技術者のことである．T型技術者をめざそう！

まずはT型技術者を目ざそう

「I型技術者」と「T型技術者」ということばを聞いたことがあるでしょうか（図1）．

I型技術者とは，自分が担当している分野ではほかを寄せ付けないほど非常に深い知識を持っている人，つまりその道を極めた技術者です．一方，T型技術者とは，自分が担当している分野だけでなく，その周辺にある技術に対しても幅広い知識を備えた技術者です．

● 技術の変化への対応力が必要

筆者はT型技術者になりたいと努力し続けています．その理由は，自分が育った時代背景にあります．

筆者が青春を過ごしたのは1970年代で，オイルショックや円高不況など，大きな景気の波に何度も襲われました．大学進学時の景気は最悪で，企業の採用も極端に落ち込んでいました．安定した公務員になろうとする傾向が非常に強いなか，進学先に電子系の学部を選択したところ，「就職のあてがない電子系に行くなんて，学費をどぶに捨てるようなものだ」とさえ言われました．ところが大学生活が始まると景気が徐々に戻り始め，パソコンの登場によりエレクトロニクス業界が活性化されました．筆者が卒業するころには世の中全体がディジタル化へ進み始め，その流れに乗ったまま今のしごとに就きました．一方，当時は華やかだったのに今は見る影もなくなった産業も数多くあります．

自分は運が良かったということを言いたいわけではありません．一度は進もうかとも考えた分野で職を失った方たちをまのあたりにしたことにより，「あすはわが身」という意識で自分が今立っている場所の風の流れを読み，必要に

応じて自分の場所を移動することが重要です．筆者は，そのときそのときの製造技術ぎりぎりのチップ・サイズとなるほど大規模な製品を，仕様策定から数えると数年もの期間をかけて開発していたことがあります．その後，何か新しい目標を求めるようになりました．幸いにも社内には組み込みシステムの開発部門があり，ASICやFPGAの開発を通じてシステム開発を担当する機会を得ることができました．自分が設計したものを製品として出荷するまでの厳しさとともに，それを乗り越えて出荷にこぎつけたときの喜びを味わいました．FPGA/ASIC＋マイコンという組み込みシステムの設計を担当したことで，システムLSIの時代になっても，何の不安もなく開発に取り組むことができています．ここ数年はFPGAとソフト・マクロのCPUを組み合わせた設計をおもに担当しています．LSI設計だけを担当していたらおそらく味わえなかったであろう貴重な経験を数多くできたことは，大きな財産です．このように，「つぶしのきく技術者になる」ことは，変化の激しい技術の世界ではたいせつなことです．

専門知識や技術はいつでも必要とされていますが，その内容は生き物のようにどんどん変化しています．このことに気がつかないと，いつのまにか，使われなくなった技術だけを持った化石のような技術者になってしまいます．

● LSI設計で求められるスキル

LSI設計におけるT型とはなんでしょうか．字の形を穴に例えると，縦穴は言うまでもなく，担当分野の専門技術です．

1) 仕様を正しく理解して，最適な論理構成を選択できること
2) 不ぐあいのない論理設計ができること
3) 設計支援ツールを使いこなせること
4) 不ぐあい発生時の解析が的確で，早期に原因を特定し対策できること
5) ドキュメントがていねいで，ほかの技術者への移管が容易なこと

順不同ですが，おもにこのような内容です．もちろんI型もこの縦穴であることはおわかりだと思います．

次に横穴に相当する関連技術の例です．

1) 論理設計以外のLSI開発工程に関する知識を持つこと
2) 基板レベルの設計技術を持つこと
3) ソフトウェアの設計技術を持つこと
4) 開発するLSIがターゲットとしているアプリケーション分野に関する知識を持つこと

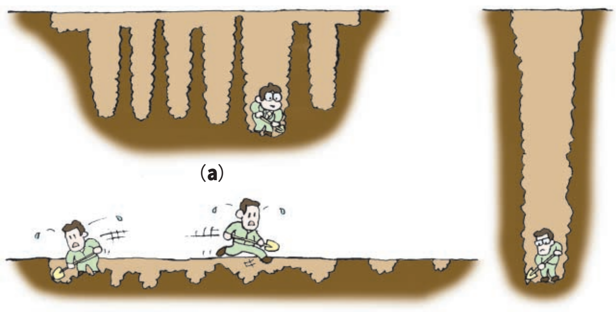


図2　経験を積み重ねてめざす技術者

(a)は，究極の目標の一つである複数の縦穴を持った「くし型技術者」．一つの技術に対し十分な知識を身に付けた後，新しい分野にチャレンジすることを繰り返してキャリア・アップし，2本目，3本目の縦穴を掘る．(b)のように世界の第一人者と認められるまで縦穴をひたすら掘り進むのも一つの道．(c)のような横穴だけというのは単なる雑学．

究極の目標の一つは複数の縦穴を持った「くし型技術者」になることです（図2）．一つの技術に対して十分な知識を身に付けた後，新しい分野にチャレンジすることを繰り返してキャリア・アップします．このとき，2本目の縦穴を作るにしても横穴がないと始まりません．あるいは，世界の第一人者と認められるまで縦穴をひたすら掘り進むのも一つの道かもしれません．横穴だけというのは単なる雑学で，技術とは呼べません．

● 現実とのギャップをなくす

HDLによる設計が中心となるにつれて，基礎技術であるはずのゲート・レベルの考えかたは，学習（歴史？）的要素が強くなってしまいました．会社や教育機関などでは最初から教えていないというケースもあるようです．

さらに，開発サイクルがどんどん短くなることで，LSI設計の専業（分業）化が進みました．自分が担当している前後の工程に関する知識がなくても，とりあえず不自由しません．しかし，基礎技術を理解せずに深みを増すことなどできませんし，すぐ届く距離にある技術に手を出さない技術者は，刻むのを止めた時計といっしょです．

システムLSI開発では，一つのLSIがシステム全体の機能を包含するわけですから，システム設計の技術が求められます．にもかかわらず，LSIの外側（使われかた）からではなく，内側（作り勝手）からしかものを見ることができないのでは，ほんとうに良い製品を作り出すことなどできるわけがありません．

今後，Cベース言語によるハードウェア設計が本格的に

普及し始めれば，アプリケーション全体がC言語で記述され，その中からどこをLSI化すれば最適なシステムが生まれるかを見極める技量が求められるようになるでしょう．LSI化する部分の切り出し終了＝LSIの設計終了，後はコンパイラまかせというのは極端かもしれませんが，コンパイラ技術は日々進化しているので，まったくの夢物語ではないと思います．

LSI設計という分野で技術者の世界に飛び込んだ限りは，縦穴を深くすることが何よりも重要ですが，いつまでもそこに留まることなく，ときには目線を変えて横穴を広げていってください．

● LSI設計者の壁

LSI設計者になると，だれもがかならず一度は思うことがあります．

「自分が設計を担当したLSIは，どこでどのように使われているのだろうか？」

筆者が社会人になってしばらくたったとき，大学の仲間と会う機会がありました．そして自然と，今どんなしごとをしているのかという話になりました．

「この前発売されたあの製品知ってる？ 実はあの製品の一部はオレが設計したんだ」と自慢して話すことができれば，モチベーションを高く保ってしごとができるでしょう（図3）．しかし，筆者が設計していたLSIは汎用品に分類されるものでした．決してマイナなLSIではありませんでしたが，どこでどのような製品に搭載されたかと聞かれてもほとんど答えられませんでした．社内に機器設計の部門があり，社内向けのLSIを開発しない限り，製品に組み込まれた形で実際に動作しているところを見ることはほとんどないのが現実です．このようなとき，“部品メーカの技術者”としてのジレンマを強く感じたものです．

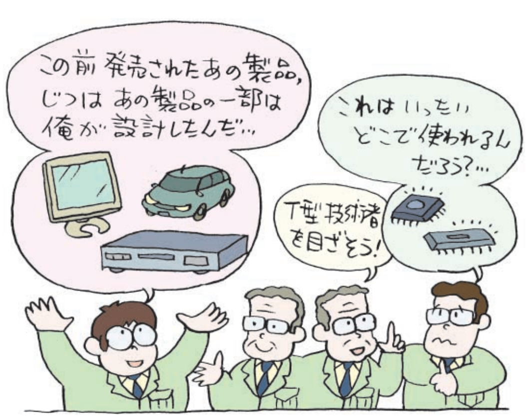


図3　モチベーションを保つ

自分が設計を担当したLSIは，どこでどのように使われているのだろうか？ 自分が開発したLSIが使われた製品が身近にあると，しごとに対するモチベーションを保ちやすい．実際に動作しているところを見ることができないと，“部品メーカの技術者”としてのジレンマを強く感じてしまいがちである．このような場合は，T型技術者，くし型技術者になることそのものを目標にするとよい．

LSI設計というしごとで，達成感をどのようにして得るのかを自分なりに悩んだ結果，T型技術者になることが最善なのだと気がつきました．担当範囲の中だけを掘り下げるのではなく，T型技術者になることで自分の担当範囲の位置づけを理解し，自分が設計したLSIがどのような形で使われるのかをユーザの立場で見るようにすることが最善なのだと気がつきました．

システムLSI開発が主流となった今では，単なるT型のLSI設計者ではなく，LSI開発のできるシステム設計者を目ざしています．

基礎を身に付けよう

まず，論理設計に対してどれだけの知識/経験を持っているのかを自己検証していただこうと思います．表1の質問に関して該当する項目をチェックしてください．

● 設計技術力の自己検証

表1の質問は，チェックの付く場所がいくつかのパターンに分かれるように作成したつもりです．チェックの付きかたによって，現状のレベルと今後の努力目標をまとめます．

1）まったくチェックがない

LSI設計者になろうとしていて，このリストにまったくチェックが付かない方はいないことを期待しています．もしチェックが付かなければ，あなたが飛び込もうとする世界が，ものすごく奥深い技術のうえに成り立っていると認識することが重要です．そして今のうちに論理設計の入門

表1　論理設計知識/経験チェック・リスト

No.	質　問	チェック欄
1	論理合成ツールやシミュレーション・ツールを使える	
2	言語設計の経験がある	
3	論理合成ツールが自動生成した論理回路を見たことがある	
4	ゲート・レベルで論理回路を設計したことがある	
5	論理圧縮の意味や手法を知っている	
6	自分で論理圧縮をしたことがある	
7	ブール代数を知っている	
8	10進/2進/8進/16進数間の数値変換や演算ができる	

書を読み，ディジタル回路の基本となる2進数('1'/'0')の世界に足を踏み入れてください．

やる気さえあれば心配する必要はありません．ディジタルに関する知識を持ってこの世に生まれてくる人間はどこにもいません．つまり，最初はだれでもチェックが付けられない状態から始まるのです．

筆者の場合は，中学生くらいまでは完全にアナログの世界であり，ディジタルの世界を知ったのは高校生になってからでした．インターネットやパソコンなどを使う今の中高生よりも，ディジタルについての知識がなかったでしょう．そして，当時ちょっとしたきっかけでディジタルの世界を知って足を踏み入れ，興味を持った人の一部が，今技術者としてしごとをしているわけです．

2）7番以降にチェックがある

ディジタル論理の基礎である2進数に関する知識があるので，この世界に違和感は持たないでしょう．これからは，学生時代に学んだ知識を実践できるようになることがたいせつです．

演習問題1

表2に示す真理値表を簡単にしてください．

(p.26のコラム「ブール代数による式の簡単化 ― 演習問題1の解説」を参照)

3）5番以降にチェックがある

ディジタル論理の基礎知識は十分に持っています．次はゲート単位ではなく，機能的な論理について理解していただきたいと思います．

演習問題2

4個の2入力NANDで2入力XORを作ってください(図4)．

(p.27のコラム「4個の2入力NANDで2入力XORを作る ― 演習問題2の解説」を参照)

演習問題3

2進数演算の基本要素である全加算器(full adder)を設計してください．

(pp.28-29のコラム「全加算器を設計する ― 演習問題3の解説」を参照)

4）4番以降にチェックがある

LSI設計の世界に入るにもかかわらずまだHDLを使ったことがないというのは，小規模な設計しか経験がないからでしょう．

HDLによる設計は，デバイスやツールへの依存性を少なくするだけでなく，設計資産の再利用が容易になるなどのメリットがあります．最近は無償で利用できるPLD開発ツールでもHDLをサポートしているので，ぜひチャレンジしてみましょう(図5)．

5）2番だけにチェックがある

HDLのコーディングだけしかやったことがないという人はあまりいないと思います．もしいたとしたら，単なる言語としての知識ではなく，自分の技術として定着するようにがんばりましょう．

6）1番と2番だけにチェックがある

即戦力として扱われるかもしれません．しかし，論理回路は多くの基礎技術の上に成り立っていることを認識してください．論理合成結果の回路を眺めることあたりから始めて，周辺技術を勉強すると，論理設計者としての幅を広げることができます．

7）散発的にチェックがある

技術レベルを意識して質問を並べたつもりなので，チェックが散発的に付くことはあまりないと思います．散発的になった方は，間を埋めるように努力しましょう．

● 実習主義で基礎を学ぼう

世の中のすべてにおいて基礎は大事です．基礎がしっかりしていないと，その上にどれほどすばらしいものを築いても，足下から崩れてしまいます．

ディジタル論理のLSIは，AND，OR，XORなどのゲート論理，およびフリップフロップに代表される記憶素子の組み合わせで実現されています．これらの知識がないと話になりませんが，教科書を読んでもなかなか身に付かない

表2 演習問題1 ― この真理値表を簡単にする

入 力			出 力
A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

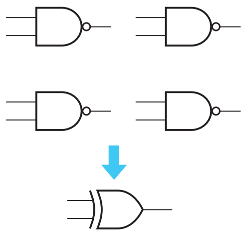


図4 演習問題2 ― 4個の2入力NANDで2入力XORを作成する

ブール代数による式の簡単化 ― 演習問題 1 の解説

ここではブール代数を用いて式を簡単化した場合の例を示します．まず，真理値表中で出力Zが1になるときの入力条件をすべて並べた式を立てます．

$$Z=\overline{A}\cdot\overline{B}\cdot C+A\cdot\overline{B}\cdot\overline{C}+A\cdot\overline{B}\cdot C+A\cdot B\cdot\overline{C}+A\cdot B\cdot C \quad \text{(A-1)}$$

式(A-1)の形を主加法標準形と呼びます．次に，通常の代数と同じように同じ変数(ここではA)を持つものを洗い出し，かっこでくくります．

$$Z=\overline{A}\cdot\overline{B}\cdot C+A\cdot\left(\overline{B}\cdot\overline{C}+\overline{B}\cdot C+B\cdot\overline{C}+B\cdot C\right) \quad \text{(A-2)}$$

式(A-2)のかっこの中を見てすぐに思いつく方もいるでしょうが，わかりやすくするため，さらにかっこの中をくくります．

$$Z=\overline{A}\cdot\overline{B}\cdot C+A\cdot\left(\overline{B}\cdot\left(\overline{C}+C\right)+B\cdot\left(\overline{C}+C\right)\right) \quad \text{(A-3)}$$

式(A-3)の中で，いちばん内側のかっこ内の$\overline{C}+C$は1ですから置き換えます．

$$Z=\overline{A}\cdot\overline{B}\cdot C+A\cdot\left(\overline{B}\cdot 1+B\cdot 1\right) \quad \text{(A-4)}$$

式(A-4)の1を省略します．

$$Z=\overline{A}\cdot\overline{B}\cdot C+A\cdot\left(\overline{B}+B\right) \quad \text{(A-5)}$$

式(A-3)と同じように，$\overline{B}+B$は1ですから，置き換えて1を省略します．

$$Z=\overline{A}\cdot\overline{B}\cdot C+A \quad \text{(A-6)}$$

一見，きれいになったように見えますが，まだ終わりではありません．ここで式(A-7)に示すブール代数の定理を思い出してください．

$$X+\overline{X}\cdot Y=X+Y \quad \text{(A-7)}$$

式(A-7)を，

$$X=A,\ Y=\overline{B}\cdot C$$

として式(A-6)を置き換えます．

$$Z=\overline{B}\cdot C+A \quad \text{(A-8)}$$

これで簡単化は完了です．式(A-8)は**表2**をもっとも簡単化した式，言い換えれば最小まで圧縮した論理です．現在，HDL設計で活用されている論理合成ツールも，組み合わせ回路に対する圧縮方法は基本的にこれと同じです．多入力多出力の組み合わせ回路に対しても，与えられたゲート数や遅延時間などの条件下で最適な論理を生成するように作られています．



(a) Quartus II Web Edition



(b) ISE WebPACK

図5 無償で使えるPLD開発ツール
(a)は米国Altera社の「Quartus II Web Edition」，(b)は米国Xilinx社の「ISE WebPACK」．

ものです．それではどうやって基礎技術を学べばよいのでしょうか．

筆者は自分で設計/製作する実習主義で体得してきました．74シリーズというTTL(transistor transistor logic)の汎用論理ICを購入し，自分で設計した回路をユニバーサル基板上に実現し，7セグメントLEDを点灯させたりして，まず作る楽しさを覚えました．それと並行して電話帳ほどの厚みがあるデータ・ブックを購入しました．そして，論理的な動作や電気的仕様の読みかたを学び，知識を付けるようにしていきました．紙から学ぶだけでは飽きてしまうタイプの方には，この方法がお勧めです．部品代や買い付けのための交通費など自己投資が大きい分，むだにしたくないという良い方向のプレッシャも加わるので効果抜群です．

当時は個人でオシロスコープを所有することは難しく，テスタ程度しか持っていませんでした．このような環境におけるデバッグはものすごくたいへんでした．しかし，う

4個の2入力NANDで2入力XORを作る ― 演習問題2の解説

表B-1にNANDとXORの真理値表を示します．まずXORの論理式を立てます．

$$Z=\overline{X}\cdot Y+X\cdot\overline{Y} \qquad \text{(B-1)}$$

式(B-1)をこねくり回していっても解が求められますが，ちょっと違った方法でアプローチしてみます．

まず，式(B-1)をそのまま回路図にすると図B-1になります．2入力NANDをまったく使っていませんが，2入力のゲートが3個，そのうちAND(論理積)が2個あります．論理設計の初心者でも勘の良い方ならば，これらの2入力ゲートがNANDにできると予想できるでしょう．

ANDとORをNANDにするには，ブール代数の定理を使います．

$$\overline{A+B}=\overline{A}\cdot\overline{B}$$
$$\overline{A\cdot B}=\overline{A}+\overline{B} \qquad \text{(B-2)}$$

$$\overline{\left(\overline{A}\right)}=A \qquad \text{(B-3)}$$

式(B-2)はドモルガン(De Morgan)の定理と呼ばれ，非常に重要です．式(B-1)で，

$$A=\overline{X}\cdot Y,\ B=X\cdot\overline{Y}$$

とします．

$$Z=\overline{X}\cdot Y+X\cdot\overline{Y}=A+B \qquad \text{(B-4)}$$

式(B-4)に式(B-2)の定理を適用します．

$$Z=A+B=\overline{\left(\overline{A}\cdot\overline{B}\right)} \qquad \text{(B-5)}$$

AとBを元に戻すと，論理積だけの式となります．

$$Z=\overline{\left(\overline{A}\cdot\overline{B}\right)}=\overline{\left(\overline{\left(\overline{X}\cdot Y\right)}\cdot\overline{\left(X\cdot\overline{Y}\right)}\right)} \qquad \text{(B-6)}$$

式(B-6)をそのまま回路に直したものが図B-2です．出力段にあるNANDを前述のドモルガンの定理でシンボル変換したものを図B-3に示します．この変換を各ノードの論理合わせと呼びます．あまり意味がないように思われるかもしれませんが，後で論理図を追いかける際には論理を合わせておいたほうがわかりやすくなります．

ようやく3個のNANDを使うことができました．あと一歩です．残りの2個のNOTをどのようにしてNANDへ置き換えればよいのでしょうか．2入力2出力回路となり一見不可能に見えますが，式(B-6)を下記のように展開していきます．

$$\begin{aligned}Z&=\overline{\left(\overline{\left(\overline{X}\cdot Y\right)}\cdot\overline{\left(X\cdot\overline{Y}\right)}\right)}=\overline{\left(\overline{\left(\overline{X}\cdot Y+Y\cdot\overline{Y}\right)}\cdot\overline{\left(X\cdot\overline{X}+X\cdot\overline{Y}\right)}\right)}\\&=\overline{\overline{\left(Y\cdot\left(\overline{X}+\overline{Y}\right)\right)}\cdot X\cdot\overline{\left(\overline{X}+\overline{Y}\right)}}\\&=\overline{\left(\overline{Y\cdot\overline{\left(X\cdot Y\right)}}\cdot\overline{X\cdot\overline{\left(X\cdot Y\right)}}\right)}\end{aligned} \qquad \text{(B-7)}$$

式(B-7)は，4個の2入力NANDの演算式でできています．わかりづらい方は，図B-4と対応させてみてください．式(B-7)でいちばん内側の，

$$\overline{\left(X\cdot Y\right)}$$

が図B-4のノード①の出力，同様に，

$$\overline{\left(Y\cdot\overline{\left(X\cdot Y\right)}\right)}$$

がノード②の出力，

$$\overline{\left(X\cdot\overline{\left(X\cdot Y\right)}\right)}$$

がノード③の出力，そしてノード④の出力がXORの出力となります．

論理設計において，最適解はかならずしも一つではありません．機能はまったく同じでも，使えるゲートの品種や許される遅延時間など，与えられる条件に応じた最適化を行うことが必要です．

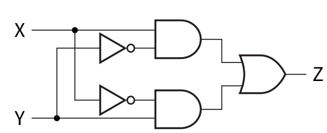
表B-1　NANDとXORの真理値表

入　力		出　力
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

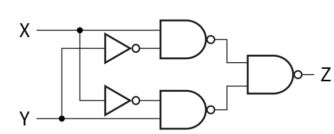
(a) NAND

入　力		出　力
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

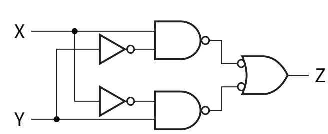
(b) XOR



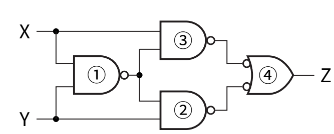
図B-1　式(B-1)をそのまま回路図に



図B-2　式(B-6)をそのまま回路図に



図B-3　3個のNANDと2個のNOTならできるのに…



図B-4　4個のNANDで構成したXOR回路

まく動かないときの現象解析方法や，回路図に現れない電源の引き回しかた，さらには効果的なパスコンの入れかたなどを体験を通じて身に付けることができました．そして何よりも自分で設計した回路が動いたときの喜びが，今の

1

全加算器を設計する ― 演習問題３の解説

全加算器の設計は，まず半加算器（half adder）を知ることから始まります．表C-1に半加算器の真理値表を示します．２本の入力に与えられる‘１’の数を数え，10進の０～２に対応する“00”，“01”，“10”という２進２けた（２ビット）の数値として表現（エンコード）します．出力HCは半加算によるけた上げ信号（half carry），出力HSは半加算によるそのけたの演算結果（half sum）と呼ぶことにします．

図C-1に半加算器の論理回路を示します．HCはAND回路，HSはXOR回路で実現できます．

次に，半加算器を使って全加算器を構築することを考えます．表C-2に全加算器の真理値表を示します．全加算器は，半加算器の２本の入力に，一つ下のけたからのけた上げ入力を加えた加算器です．３本の入力に与えられる‘１’の数を数え，10進で０～３に対応する“00”，“01”，“10”，“11”という２けたの２進数（２ビット）の数値として表現（エンコード）します．出力のSumはそのけたの演算結果，Carryは一つ上のけたに対するけた上げになります．

どうやって半加算器を使って全加算器を構成できるのでしょうか．

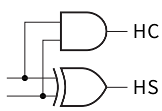
まず，表C-2で入力A＝‘０’の部分（真理値表の上側半分）だけに注目してみます．入力Aは変化しないので見ないようにすると，表C-1で示した半加算器の真理値表とまったく同じであることに気がつくでしょう．

次に，A＝‘０’のときの入力BとCに対する出力と，A＝‘１’のときの入力BとCに対する出力を比較してみます．するとA＝‘０’のときの出力に対して，A＝‘１’では‘１’加算された値となっていることがわかるはずです．どうやら半加算器を２個使い，初段の半加算器でまず２入力分の演算出力（HS）を得て，その出力と残り１本の入力とを半加算器で演算すれば，そのけたの演算結果（Sum）は作れそうです（図C-2）．

次に，２本あるけた上げ出力（HC）をどのようにまとめたら全加算器のけた上げ出力にできるかを考えます．図C-2と表C-2をもう一度

表C-1　半加算器の真理値表

入　力		出　力	
X	Y	HC	HS
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



図C-1　半加算器の論理回路

表C-2
全加算器の真理値表

No.	入　力			出　力	
	A	B	C	Carry	Sum
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

自分の原点になっていると信じています．

実際に入社２年目で，標準論理ICを約50個搭載した基板を３枚使用するシステムの仕様作りから設計，製作，デバッグまでをひとりでこなし，期限までに完全動作させたときはものすごく自信を持ちました．

それから20年以上経過していますが，最近，また物づくりの虫が騒ぎ出しています．個人でも購入できる価格帯になったこともあり，ディジタル・オシロスコープや直流安定化電源などを買いそろえました．現在，さらに信号発生器（signal generator）を物色中です．

● しごとの世界に興味を持とう

筆者は，偶然にも学生時代に興味を持った分野のしごとに携わっており，非常に恵まれていると思います．しかし，長い社会人生活の中では，設計以外のしごともいろいろとやってきました．そのようなときにも，「好きこそもののじょうずなれ」の精神で，まずは興味を持って好きになるように努力し，それから一生懸命取り組みました．そのスタイルを貫けたからこそ今があるのだと信じています．

現在は，回路図エディタのシンボルでしかゲートを見ることがなく，デバッグもシミュレーションが中心の，いわゆるバーチャルな設計の時代です．そのため，作る（動かす）喜びを実感したことのない方がいるかもしれません．しかし最近では，FPGAボードやマイコン・ボードが付属している雑誌や書籍があります．ポケット・マネーで購入できる数千円程度のものなので，自分で購入し，そしてとにかく動かしてみましょう．

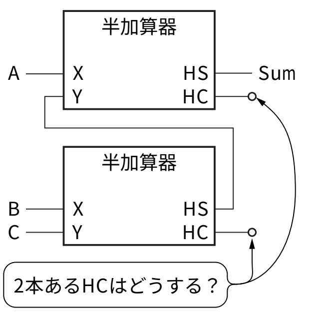
最初は動かない→なぜ動かないかを考える→対策して動くようにする→動かして楽しむ→興味がわいた部分を深く掘り下げて勉強する，というステップをたどればしめたものです．すべての人に最善であるとは言えないかもしれませんが，強い情熱を持ってしごとをする人間を少なくとも１人は生み出した方法であるということを，その本人（筆者）が保証します．

よく見て，それぞれのHCが‘1’となる条件を考えます．すると，入力BとCを入力する初段の半加算器はB＝C＝‘1’(状態No.3とNo.7)のときにHC＝‘1’となり，次段の半加算器はA＝‘1’のときでかつBとCのいずれか一方が‘1’(状態No.5とNo.6)のときにHC＝‘1’となります．そうです，この二つのHCは同時には‘1’とならないので，単純に論理和をとれば，全加算器のCarryとすることができるのです．

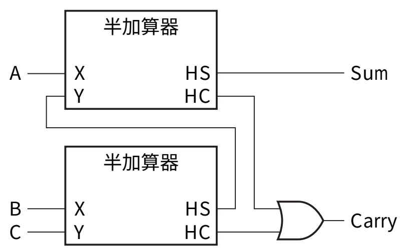
完成したのは2進数1けたの全加算器ですが，Carry出力を上位けたの入力として接続することで，何けたの加算器でも実現することができます．このような構成の加算器をリプル・キャリ方式の加算器と呼びます．ただし，図C-4に示すように，リプル・キャリ方式の加算器は，ビット数に比例して上位けたのキャリを生成するために通過するゲート段数が増えていきます．つまり，演算結果が得られるまでの時間が長くなります．それを回避するため，上位けたへのキャリを少ないゲート段数で生成するキャリ・ルックアヘッド方式などの高速演算方式が考案されています．

真理値表の簡単化による論理設計手法では，多ビット入力の論理回路を設計することは容易ではありません．かならず機能的にとらえて，よりマクロなレベルで設計することが求められます．

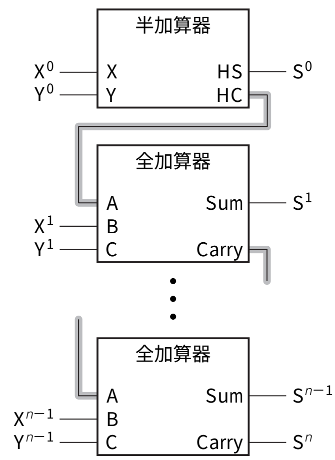
実務ではHDL設計が中心にはなると思いますが，表1の3番と4番にもチェックが付くように研さんを積めば，どこに出ても負けない論理設計者になれる可能性があります．



図C-2　全加算器のSum部分の回路



図C-3　完成した全加算器



図C-4　リプル・キャリ方式の加算器

● HDL設計は新しい基礎

HDL＋コンパイラ(論理合成)の出現により，LSIの設計スタイルは大きく変化しました．ちょうどソフトウェアがアセンブリ言語からFORTRANやC言語などの高級言語へ移行したのと似ており，生産性の向上による設計期間の短縮は目を見張るばかりです．設計と製造の両面の技術発展がなければ，今のLSIは生まれていません．無理だろうと言われていた壁を，技術は何度も乗り越えてきましたし，目標がある限りは今後も乗り越え続けるでしょう．

回路図をベースとするゲート・レベル設計時代の基礎はブール代数でした．現在のHDL設計は，その上に新しい基礎を築いています．興味を持つためには，わかりやすいことがたいせつですから，まずはHDL設計を身に付けてください．そしてさらに，ぜひゲート・レベルの設計技術もマスタし，深みのある設計者を目ざしてください．

● EDAツールに負けるな

LSI設計では，EDAツールに接する時間が多い分，その操作に慣れることができないと，設計者が感じるストレスの量ははんぱではありません．ツールはその名のとおり道具です．プラットホームのパソコンやワークステーションも含めて使いこなすことが重要です．おっかなびっくり操作していたのではやる気が起きませんし，逆にいや気がさしてしまいます．

筆者はこれまでに何台ものパソコンを自分で購入し，現在も複数台が稼働中のため，大のパソコン好きと見られています．しかし，実際にはパソコンなんか大嫌いです．しごとで使う必要がなければ，見向きもしないだろうと思います．そんな人間だからこそ，「使ってやる」と自分が見下ろすことで精神的に優位に立てるほど使い込むのです．

ソフトウェアのインストールやセットアップを他人任せにしていませんか？ もしそうなら，そのあたりから克服していき，EDAツールもこき使ってやりましょう．機械に負けるな！

あさい・たけし　日立エンジニアリング(株)